コンピュータによる連番の付け直し

―言語学文書作成支援プログラム―

田野村　忠　温

1　はじめに

言語学関係の文書――論文や書物、学会・研究会での発表資料などを「文書」と総称することにする――においては、議論の必要上、多数の例文を示すことが多い。そのようなとき、すべての例文に連番（通し番号）を与えておき、議論や説明にあたってはその連番を用いて必要な例文を参照するということがしばしば行われる。例文だけでなく、その他の種類の事例、仮説や条件、結論の要約などを提示・参照するときに、同様に連番を用いることも多い。

連番の使用は便利である一方で、１つだけ不便な点がある。それは、いったん文書を書き上げた後で、例文その他を追加したり、削除したり、順序を変更したりといった修正を施すと、連番を文書全体にわたって付け直さなければならなくなるということである。(1)

この連番の付け直しの作業は、例文その他の冒頭に付けられた連番を付け直すだけでも少なからぬ時間と労力を必要とするものであるが、多くの場合、議論の中で例文を参照するために用いた連番も同時に付け直さなければならないため、その時間と労力はたいへんなものになる。そして、その結果として、連番の付け直しを忘れたり、間違えたりするという危険を伴うことにもなる。

2　連番の付け直しの自動化

連番の付け直しの作業はこのようにやっかいなものであるが、そのアルゴリズム自体はまったく単純なものである。そこで、文書をコンピュータ（パソコンを含む）やワープロで作成しているのであれば、このような作業はコンピュータに任せてしまうということが考えられる。事実、そうすることは可能であり、また、そうするのが合理的である。

こうした考えから、筆者は、パソコン上で文書中の連番を自動的に付け直すプログラムを開発し、以来、文書の作成に際してはそれを利用している。これにより、文書の作成に要する時間と労力を節約することができるし、当然、その節約できる時間と労力は文書の質そのものの向上に充てることができることになる。

このプログラムを使えば、例えば、（図１）に示すような文書は、自動的に（図２）のように変換される。

（図１） 処理前の文書

§1　不連続の連番を連続に

次に示す例文のうち、(5)と(10)は受動文で、それぞれ、(3)と(8)の能動文に対応している。

(3) 太郎は花子を愛している。

(5) 花子は太郎に愛されている。

(8) John loves Mary.

(10) Mary is loved by John.

なお、これと同じ例文を§2でも用いる。

§3　連番の大小の順序の逆転もＯＫ

次に示す流行歌の歌詞のうち、(13)～(9)は昭和20年代の歌、(2)は昭和30年代の歌の冒頭部分である。

(13) 白い夜霧の灯りに濡れて 別れ切ない プラットホーム（注2）

(9) 粋な黒塀見越しの松に 仇な姿の洗い髪（注1）

(2) 泣けた泣けた こらえきれずに泣けたっけ（注5）

§2　a,b,c などの付いた連番もＯＫ

次に示す例文のうち、(55b)と(24b)は受動文で、それぞれ、(55a)と(24a)の能動文に対応している。これらは§1で用いた例文である。

(55) a. 太郎は花子を愛している。

b. 花子は太郎に愛されている。

(24) a. John loves Mary.

b. Mary is loved by John.

（注2）春日八郎(1952)「赤いランプの終列車」

（注1）春日八郎(1954)「お富さん」

（注5）春日八郎(1955)「別れの一本杉」

（図２） 処理後の文書

§1　不連続の連番を連続に

次に示す例文のうち、(2)と(4)は受動文で、それぞれ、(1)と(3)の能動文に対応している。

(1) 太郎は花子を愛している。

(2) 花子は太郎に愛されている。

(3) John loves Mary.

(4) Mary is loved by John.

なお、これと同じ例文を§3でも用いる。

§2　連番の大小の順序の逆転もＯＫ

次に示す流行歌の歌詞のうち、(5)～(6)は昭和20年代の歌、(7)は昭和30年代の歌の冒頭部分である。

(5) 白い夜霧の灯りに濡れて 別れ切ない プラットホーム（注1）

(6) 粋な黒塀見越しの松に 仇な姿の洗い髪（注2）

(7) 泣けた泣けた こらえきれずに泣けたっけ（注3）

§3　a,b,c などの付いた連番もＯＫ

次に示す例文のうち、(8b)と(9b)は受動文で、それぞれ、(8a)と(9a)の能動文に対応している。これらは§1で用いた例文である。

(8) a. 太郎は花子を愛している。

b. 花子は太郎に愛されている。

(9) a. John loves Mary.

b. Mary is loved by John.

（注1）春日八郎(1952)「赤いランプの終列車」

（注2）春日八郎(1954)「お富さん」

（注3）春日八郎(1955)「別れの一本杉」

この変換例が示しているように、例文などの番号だけでなく、章・節の番号、注の番号なども自由に付け直すことができる（それぞれを別々に処理する）。処理に要する時間は使用するハードウェアの条件――主に、ＣＰＵの処理速度と、ディスク装置の入出力の速度――に依存するが、例えば400字詰め原稿用紙に換算して100枚程度の文書であれば、いくら訂正箇所が多くても数秒

程度で処理が完了する。

以下においては、コンピュータによる連番の付け直しの手順と、それを実際にプログラミングするうえで問題となってくる点について解説する。なお、筆者の作成したプログラムはフリーソフトウェアとして一般に公開しているので、自由にご利用いただくことができる（５節参照）。

３　コンピュータによる連番の付け直しの手順

すでに述べたように、連番の付け直しのアルゴリズムは、まったく単純なものである。小学校の算数程度と言ってよいであろう。したがって、これをプログラムとして実現することは、ある程度のプログラミングの心得があれば、むずかしいことではない。

ここでは、プログラミングの初心者で、連番を付け直すプログラムを自分で作ってみようと思われる方や、プログラミングは行わないが、どのようなものか知りたいと思われる方を念頭に置き、コンピュータによる連番の付け直しについて解説することにする。

３．１　基本方針

プログラムをどのように組み立てるかということに関しては、当然、人と場合により様々な可能性があり得る。しかし、連番を付け直すプログラムを開発するうえで、特に重視するに値すると筆者が考えた方針が１つだけある。

それは、まったく普通の形式の文書を処理できるようにするということである。つまり、論文なら論文としてそのまま印字できるような体裁の文書を用意しておけば、それをプログラムで直接に処理できるようにするということである。

もしも、例えば、連番であることを示す特別な記号などをあらかじめ文書の中に埋め込んでおかなければならないとすると、文書の作成によけいな手間がかかることになり、間違いも起こりやすい。また、印字の段階では、今度はその記号を除去する作業が必要となる。さらに、いったんその記号を除去してしまうと、二度と連番を付け直すことができなくなってしまうという、重大な問題点もある。

他方、プログラムが通常の形の文書を処理できるようにしておけば、何度でも繰り返して連番を付け直すことができ、しかも、任意の時点で文書をそのまま印字することができることになる。

ワードプロセシングの形態によっては事情が異なってくる可能性もあろうが、少なくとも筆者としては、そのような方針に従うのがもっとも利用しやすくてよいと考え、それに従った。

３．２　用語の定義

さて、具体的な説明に入る前に、いくつかの用語を定義しておく必要がある。

まず、「行」とは、改行によって区切られた文字の連続とする。「段落」とか「論理行」など

と呼ばれるものに相当する。画面上の表示の単位やプリンタへの印字の単位ではないことに注意していただきたい。

「行頭」とは、基本的には、当然のことながら、行の１字目からの位置を言うが、特に、行が空白文字（の連続）で始まる場合は、その空白文字（の連続）を除いて得られる行の１字目からの位置を行頭と呼ぶことにする。

次に、「連番」とは、「(5」「(23」などのように、左括弧「(」と、１つ以上の数字（「0」～「9」）の連続とが、この順に連続しているものと定義する。また、その数字の連続が表している数自体――つまり、５、23という数――を連番と呼ぶこともある。(2)

そして、連番――最初の意味での連番――のうち、行頭の位置に現われたものを「連番の定義」、それ以外の位置に現われたものを「連番の参照」と呼ぶ。要するに、文書中で、連番付きの例文を掲げているところが連番の定義であり、本文においてそれを参照しているところが連番の参照に当たるわけである。

３．３　連番の付け直し

ここでは、（図３）のようなごく短い文書を処理することを考える。

（図３）　処理前の文書

□次に示す例文のうち、(7)と(1)は受動文で、それぞれ、(2)と(4)の能動文に対応している。↵
□□(2) 太郎は花子を愛している。↵
□□(7) 花子は太郎に愛されている。↵
□□(4) John loves Mary.↵
□□(1) Mary is loved by John.↵

（図４）　処理後の文書

□次に示す例文のうち、(2)と(4)は受動文で、それぞれ、(1)と(3)の能動文に対応している。↵
□□(1) 太郎は花子を愛している。↵
□□(2) 花子は太郎に愛されている。↵
□□(3) John loves Mary.↵
□□(4) Mary is loved by John.↵

（ただし、「□」は空白文字、「↵」は改行を表すものとする。）

３．２で述べた定義によれば、この文書は５行から成っており――７行ではないことに注意――、２行目から５行目の各行は連番の定義を１つずつ含んでいる。また、１行目は連番の参照を４つ含んでいる。

さて、この文書の連番を付け直すにあたり、まず、次のような、複数の欄（ます目）が１列に並んでいる表を作る。欄の個数が、処理できる連番の最大値になる。原理的にはいくつでもかまわないが、999個（または、999個未満）にしておくのがよい。(3)説明の便宜上、それぞれの欄の上に１からの各整数を順に記しておく。

1	2	3	4	5	6	7	8	9	10	…	999

次に、文書の最初から最後までの各行について、順次、それが連番の定義を含んでいるかどうかを調べ、含んでいる場合には次の作業を行う。すなわち、その連番がｎであり、それが、文書の最初から数えてｍ番目に現われた連番の定義であるとすると、表のｎ番目の欄に、ｍを記入する。

例えば、ここで問題としている文書の場合であれば、最初に現われる連番の定義は、２行目の２である。つまり、この２が１番目に現われる連番の定義であるから、表の２番目の欄に、１を記入する。２番目に現われる連番の定義は、３行目の７であるから、表の７番目の欄に、２を記入する。以下同様にしていくと、表は最終的に次のようになる。

1	2	3	4	5	6	7	8	9	10	…	999
4	1		3			2					

この表ができたら、いよいよ最後の段階として、文書中の連番を実際に付け直す。これには、文書中に含まれるすべての連番――今度は、連番の定義・連番の参照の別を問わない――の１つ１つについて、次の作業を行う。すなわち、その連番がｎであれば、それを、表のｎ番目の欄に記入されている数ｍに書き換える。

問題の文書の場合だと、次のようになる。文書中に最初に現われる連番は最初の行の７であるが、これを、表の７番目の欄に書かれている数、つまり、２に書き換える。次に現われる連番は同じく１行目の１であるが、これを、表の１番目の欄に書かれている４に書き換える。以下同様にして作業を続けていくと、文書は（図４）のようになる。

これで、連番の付け直しが完了したことになる。

4　プログラミング上の諸問題

以上が連番の付け直しの基本的な手順であるが、言うまでもなく、これを実用的なプログラムとして実現するには、考慮や工夫を要する点がいろいろある。各種のエラーチェックや処理の高速化は、プログラミング一般に共通する課題であるからここで述べるまでもない。以下においては、連番を付け直すプログラムに固有の問題点のいくつかについて述べる。

【定義されていない連番が参照されているときの処理】

例えば「(5)」という連番を付けられた例文がないのに、それを本文中で参照しているというような場合である。３．２で述べた手順に即して言えば、最後の段階において、文書中の「(5)」を書き直そうとするのに、表の５番目の欄には数が記入されていないという場合である。

これではどんな連番に付け直すこともできないから、エラーメッセージを出して処理を中止するようにするしかない。

【連番の定義に重複があるときの処理】

文書中で、同一の連番が２か所以上で定義されている場合の問題である。３．２で述べた手順で言えば、表に数を記入していく段階で、これから記入すべき欄にすでに何らかの数が記入されているという場合である。

これも、基本的にはエラーにして処理を中止するのが安全であるが、プログラム実行時に、それをエラーとしない選択もできるようにしておくのがよい。同じ例文を同じ連番で再び挙げることをする場合があるが、そのような文書の場合には、定義の重複をエラーとしないで処理できる必要がある。

【連番の形をしているが連番ではないものの問題】

文書中に、連番の形をしているが、連番として扱われては困るものが含まれている場合が問題となる。例えば、「ムードの形式と意味(2)」という論文の題名の中には、「(2」という表現が含まれている。このまま処理すると、「(2」も連番の参照と見なされるので、別の番号に付け直されてしまう可能性がある。

これは、プログラムの側では如何ともしがたい問題である。したがって、文書作成時に何らかの配慮をするのがよいであろう。最も簡便な方法の１つは、連番でないものについては、数字の文字種を変えて入力しておくというものであろう。つまり、連番は必ず半角数字で表現するというように決めておき、連番として処理されたくないものについては全角数字で入力するようにしておくことで、問題を回避することができる。

【段落の冒頭における連番の参照の問題】

段落がいきなり連番の参照で始まることがあり得る。つまり、「(5)に示す例に見るように～」とか「(23)の規則によると～」といった書き方で始まる段落の可能性である。これをそのまま処理してしまうと、連番の参照ではなく連番の定義と見なされてしまうので、これも何らかの対策を要する。

１つの方法は、連番の定義の現れ得る文脈を制限することであろう。「連番の定義」の定義に条件を付加し、例えば、連番が行の冒頭にあり、しかも、直前に空白文字が２個以上ある場合に

限って連番の定義と見なす――空白文字が２個以上なければ、行の冒頭であっても、連番の参照と見なす――ようにすれば、問題が解決されることになる。

連番の定義の文脈にそのような制限を加えるかどうかを、プログラム実行時に選択できるようにしておけばよい。

【種々の形をした連番の処理】

前節では、連番を、左括弧「(」に、１つ以上の数字の連続が続いているものとして定義した。この定義をわずかに変更するだけで、章・節の番号や注の番号を付け直せるようになる。

例えば、連番を、「§」という文字に、１つ以上の数字の連続が続いているものというように定義し直せば、前節での説明とまったく同様の手順によって、「§1」「§5」といった形の表現を連番として付け直すことができるようになる。

同様に、連番を、「〈図」「（注」という文字列に、１つ以上の数字の連続が続いているものというように定義し直せば、それぞれ、「〈図1」「（注5」といった形の連番を付け直せることになる。

プログラム実行時に、数字の前に現われるべき文字列――文字左括弧、「§」、「〈図」、「（注」など――を指定できるようにしておけば、１つのプログラムで任意の形の連番を処理できるようになる。(4)

【連番の付け直しの初期値】

通常は、連番を１から始めて付け直せばよい。しかし、大きな文書などで、１つの文書を２つ以上のファイルに分けて作成しているような場合には、連番の付け直しの初期値を指定できるようになっていることが望ましい。

ただし、連番の初期値の指定には上限を設け、連番を付け直したときに、最大の連番が作業に用いる表の欄の個数――３．３での説明では、999――を超えないようにしておく必要がある。と言うのは、999を超える連番ができてしまうと、次に連番を付け直すとき、それはもはや連番として扱えなくなってしまうからである。文書中で連番がｎ個定義されているとすると、1000－ｎが連番の初期値の上限となる。それを超える初期値が指定された場合は、エラーとして、処理を中止するようにすべきであろう。

【ワープロソフトで作成された文書の処理】

ワープロソフトによって事情が異なるであろうが、「一太郎」（㈱ジャストシステム）の旧バージョンを例に取ると、文字そのものはテキストファイルに、そして、その他の情報――印刷時の書式、罫線や下線、文字の種類、文字飾りなどに関する情報――は別個のファイルに保存されるようになっている。このような場合、文字の入ったテキストファイルをプログラムで処理して、

その結果として、文字数が局所的にでも変化してしまうと、再び「一太郎」で読み込んだときに、正しく表示できなくなってしまう恐れがある。

そこで、ワープロソフトで作成された文書の場合は、連番を付け直した結果、連番の桁数が変化するような場合は、エラーとし、処理を中止するようにしておくのが安全であろう。

【縦書きの文書の問題】

文書を縦書きで印字する場合は、通常、括弧や数字を含む半角文字は横に倒れた形で印字される。これはハードウェアの制約によるものでやむを得ないところであるが、作成した文書が製版・印刷にかけられるような場合には、原稿ではとりあえずそのような見苦しい形で印字しておき、それをちゃんとした体裁に訂正するよう製版の指示を加えておくことで、結果としてきれいな印刷結果を得ることができる。ちなみに、最近発表した拙文の１つは、そのような過程を経て印刷されている。(5)

5　プログラムの利用について

筆者の開発したパソコン用プログラムは、NIFTY−Serve という全国規模のパソコン通信ネット上で、「連番整序プログラム RENUMBER.EXE」という名前のフリーソフトウェアとして公開している。誰でも自由に入手できるので、ご関心のある方はお試しいただければ幸いである。(6)

このプログラムは、３節で解説した基本的な手順をもとに、４節で述べた選択可能な機能などを付加したり、処理の高速化を図ったりしたものである。

同プログラムは、NEC PC−9801シリーズのパソコンまたはその互換機種の、MS−DOS version 3.10（または、それ以後のバージョン）上で作動するものであり、その利用にあたっては、次の条件がどちらも満たされている必要がある。

- 利用者は、パソコンおよびＭＳ−ＤＯＳの操作に関する基本的な知識を有していること。少なくとも、MS−DOS の基本的なコマンドくらいは自在に使えるようでなければならない。
- 文書はテキストファイルの形で作成されていること。テキストファイルとは、MS−DOS の TYPE コマンドを使って画面上に正常に表示させることのできるファイルとお考えいただけばよい。

必要条件ではないが、「ＶＺエディタ」（㈱ビレッジセンター）というエディタソフトをお使いになっている場合は、同プログラムとともに提供される支援マクロにより、同プログラムをいっそう便利に利用することができる。

また、同プログラムとほぼ等機能のＶＺエディタ用マクロもプログラムといっしょに提供しているので、ＶＺエディタをお使いの場合には、NEC PC−9801シリーズと互換性のないパソコ

ンであってもご利用いただくことができる。

プログラムおよびマクロの詳細なマニュアルは、プログラム・マクロとともに、テキストファイルの形で提供される。ご利用の際は、それをよくお読みいただきたい。

6　おわりに

コンピュータの素人が、ささやかなプログラミングの経験について臆面もなくあれこれと述べてきた。この小文およびここで紹介した拙作プログラムが、こうした問題にご関心をお持ちの方々の一助になれば幸いである。

[註]

（1）連番の付け直しは、多くの場合、文書全体にわたって行う必要はなく、連番の訂正があった箇所以後の部分だけについて行えば十分であろう。しかし、連番の前方参照の可能性があることから、一般には、それでは不十分である。つまり、ある例文を挙げるよりも前にその例文を参照するということがあり得るため、一般には、文書全体にわたる付け直しが必要であると言わなければならない。

（2）ゼロは、連番から除外するものとする。

ここでは、文書を横書きの体裁で作成するものとの想定に立って説明を行う。パソコンのハードウェア上の制約により、ほとんどのエディタソフトやワープロソフトは横書き入力になっている。縦書きの文書の問題は、4節で取り上げる。

なお、連番の条件から、右括弧の存在は故意に考慮から外しているのであるが、このことは2つの利点を伴う。まず、「(12)」のような表現に加えて、「(12b)」とか「(12')」などといった形の表現を使用することがあるが、数字（の連続）に続く文字の如何を不問にしておけば、「(12)」と「(12b)」や「(12')」などを同等に処理することができることになる。また、後述するように、括弧以外の文字を伴う連番の処理にも融通性が与えられる。

（3）連番の最大値を高々999としておくのがよいと言うのは、言語学関係の論文で4桁の連番が使われることはほとんどないからでもあるが、同時に別の利点もある。それは、連番の形をしていても999を超えるものは無視するようにプログラミングしておけば、「金田一春彦(1953)」のような表現における文献の発表年が連番として処理されてしまうという事態を自動的に避けることができるということである。

（4）ただし、注の番号の付け直しには、ここでは説明を省略する別の問題点がある。

（5）拙論「『も』の一用法についての覚書――『君もしつこいな』という言い方の位置付け――」（『日本語学』第10巻第9号、1991年）。

（6）「連番整序プログラム RENUMBER.EXE」の入手方法は、次の通りである（1992年5月1日現在）。

同プログラムの最新バージョンは、NIFTY－Serve のフォーラム FLABO のデータライブラリ2番に、アーカイバソフト LHA（(C)吉崎栄泰氏）で圧縮して登録してある。アーカイブのファイル名は、RENUMxxx.LZH（xxx の部分はバージョン番号）である。このアーカイブには、プログラムに加えて、マクロ、マニュアル、デモ用のバッチファイルなどが同梱されている。

RENUMxxx.LZH をダウンロードするには、まずトップメニューから「go flabo」によりフォーラム

に入り（当フォーラムに未入会なら、簡単な入会手続きが必要）、続いて「lib 2」により目的のデータライブラリに入る。それ以後のダウンロードの手順については、NIFTY−Serveのアクセスガイドや通信ソフトのマニュアルをご参照いただきたい。RENUMxxx.LZHの登録番号は、キーワードRENUMBERで検索することができる。RENUMxxx.LZHがダウンロードできたら、LHAを使ってオリジナルファイルを復元する。

ちなみに、筆者は、NIFTY−ServeのフォーラムFJAMESのデータライブラリ8番では、『日本語研究文献目録・雑誌篇〔フロッピ版〕』（秀英出版、1989年）のための検索プログラムNS.EXEを、NSxxx.EXEないしNSxxx.LZHという名前で公開している。その登録番号は、キーワードNSで検索可能である。NS.EXEは、以前の拙作の検索プログラムNSEARCH.EXEの全面的な改訂版であり、NSEARCH.EXEや上記目録に付属しているKENSAKU.EXEに比べて、①多種類の検索を自動的に連続して行える、②検索結果の出力形式を自分で定義できる、③検索速度が速い、④平仮名と片仮名の区別を無視した検索ができる、⑤不完全ながらワードサーチができる、などの特長がある。こちらも併せてご利用いただければ幸いである。

なお、NIFTY−Serveでプログラムを入手することができない場合は、下記勤務先の筆者あてにフロッピ・郵送代などの実費として600円分の郵便為替を郵送していただければ、連番整序プログラムと『日本語研究文献目録』検索プログラムをフロッピディスクに収めてお送りする。

〒562　箕面市粟生間谷東8丁目1番1号　大阪外国語大学

その際は、フロッピディスクのタイプ（5インチ2HDまたは3.5インチ2HDのいずれか）をご指示いただきたい。フロッピディスクはNEC PC−9801シリーズのパソコン上でフォーマットするので、非互換機種のパソコンをお使いの場合は、データの変換が必要となる可能性がある。

（1992. 5. 12　受理）